Synchro-Phasor Data Validation and (DV&C) Project

Phase 3, Task 2

Report on

Functional Specification for the DV&C Prototype

Prepared for the
Office of Electricity Delivery and Energy Reliability,
Transmission Reliability Program of the U.S. Department of Energy
Under Contract No.DE-AC03-76SF00098

Prepared by:

Electric Power Group

Ken Martin
Jianzhong Mo

Project Manager: Dejan J. Sobajic

January 21, 2015

Acknowledgement

The work described in this report was coordinated by the Consortium for Electric Reliability Technology Solutions, and funded by the Office of Electricity Delivery and Energy Reliability, Transmission Reliability Program of the U.S. Department of Energy through a contract with Electric Power Group, LLC administered by the Lawrence Berkeley National Laboratory.

Disclaimer

Preface

Synchrophasor systems are being deployed in power systems throughout the world. As operations and system controls become more reliant on synchrophasors, it is essential that the data is correct and accurate to prevent errors in operation. Data needs to be validated to assure no errors have been introduced in communication and processing. It also needs to be conditioned with other comparisons to assure it is accurate. Validation and conditioning must be accessible to applications using the data for making timely decisions in real-time operations. The Department of Energy (DoE) has funded this project to develop and demonstrate a prototype tool to support Phasor Data Validation and Conditioning in real-time (DE-AC02-05CH11231).

As part of Phase 3, this Report documents the functional requirements of the DV&C prototype.

Table of Contents

List of Figures

Synchro-Phasor Data Validation and Conditioning Project

Phase 3, Task 2

1. Introduction

The Synchrophasor Data Validation and Conditioning (DV&C) Project sponsored by the US department of Energy and managed by Consortium for Electric Reliability Technology Solutions (CERTS) was started in December 2012. The project objectives are to develop, prototype, and test various methods for conditioning and validating real-time synchrophasor data. The project is divided into three phases.

- Phase 1: Conceptual Design and Prototype Development
- Phase 2: Prototype Demonstration
- Phase 3: Functional Specifications of the Data Validation System

In Phase 1 Electric Power Group, LLC (EPG) completed the design and prototype development to meet the Data Validation and Conditioning requirements. These requirements have been developed by EPG based on surveys, literature research, and experience in working with customers.

In Phase 2 EPG developed an Error Simulation Utility that is capable injecting errors of various types into a synchrophasor data stream played from a file of synchrophasor data. Operation of the Data Validation and Conditioning Prototype was then demonstrated using the error simulation utility. The demonstration was performed using a stream of real synchrophasor data captured from an on-line system.

Phase 3 of this project has 3 tasks:

- Task 1: Summary Report of Lessons Learned in all three tasks
- Task 2: Functional specification for the Data Validation and Conditioning Prototype
- Task 3: Review Meeting with Project Participants

This report is for the final deliverable for the project as required for Phase 3 Task 2, a functional specification of the DV&C prototype. The prototype algorithm is composed of 6 modules, each of which examines factors at stages of processing. Processing proceeds from receiving the data on the communication channel to checking the data package, parsing out the individual components, examining these components in relation to the measurements they represent, and finally relating the measured quantities to each other. The examination at each stage is described, the flags that represent the data quality problems are detailed, and any data conditioning is listed. Each process is shown in a figure and the tests and test outcomes are listed tables.

Appendix B provides a summary of all the reports provided under this project.

2. Overview and Executive Summary

This report provides functional specifications for the Data Validation and Conditioning Prototype (DV&C Prototype) created and demonstrated by EPG for this project. The DV&C prototype is divided into six modules that examine the data quality factors as they are exposed during processing. They are arranged in order of processing from receiving the overall data as a packet to examining individual data items as they are parsed from the packet. Data found to be corrupted at the initial stages levels does not need to be examined in further detail. The overall diagram is shown in Figure 1. Data and configuration comes from the data system. User entries configure and control the DV&C. The DV&C performs the validation and conditioning according to the algorithm and user set parameters. The DV&C creates a data quality flag (DQ flag) for each signal to indicate its validity and potential errors. In most cases the signal value itself is left unchanged. However in some cases there are "fatal errors" which are of a type that indicates that the data cannot possibly be usable; in these cases the data is set to not-a-number (NaN). This is a special number designation that cannot be used, which prevents the data from being used inadvertently. In some other cases, the data may be questionable but cannot be positively determined if it is good or bad. These numbers are flagged for their uncertainty and can be left to the application to decide if they should be used. Once the DV&C has examined and flagged all the data, it passes on to the output. It can be sent as fully conditioned with bad and uncertain data set to NaN or as combined data with only bad data set to NaN. The application must use the DQ flags to decide which other data to discard and which to use. The DVC also logs the errors discovered and can send the DQ flags separately.

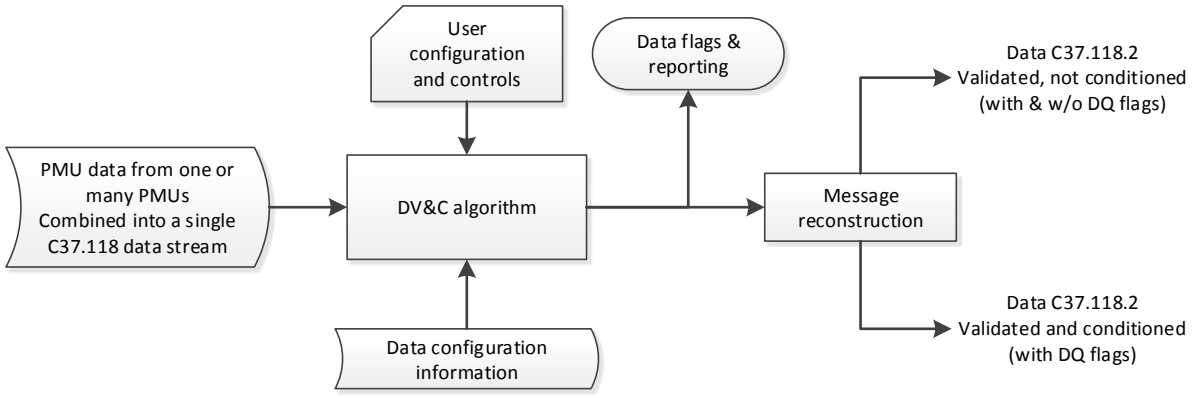

Figure 1. Basic system diagram

This system was implemented in the DV&C prototype application by EPG. It was demonstrated to the project sponsors and several other utilities at request. A summary of the actual modules and their functions are in section 3.3 below.

3. Data Validation

3.1 The data validation problem

Synchrophasor measurement was conceptualized in the 1980's and developed in the early 1990's. It showed great promise for providing the next generation measurements needed for evolving power systems. A few pilot projects followed, but without the coverage needed to make them really useful for operation and controls. After the American Recovery and Reinvestment Act (ARRA) project and other large scale initiatives, phasor measurement systems are now widely deployed in North America and throughout the world. Users of these systems are deploying applications like oscillation detection, angle monitoring, voltage sensitivity, and other stability indicators that use this data. Now with increased visibility, users are finding problems that raise concerns with data reliability. These include:

- Reliability in data delivery (significant amounts of lost data)
- Validity of magnitude and phase angle measurement
- Accuracy of the measurement
- Delays in data reporting
- Differences in values from SCADA and other measuring systems
- Measurements with excessive noise or constant values

So why do there seem to be significant problems? Do these same problems exist with other data systems? What can be done to reduce or eliminate the problems?

The central question is *"How do we guarantee that synchrophasor data meets the needs of applications that it is intended to serve?"*

3.2 Data problem discussion

Synchrophasor data systems use the same basic measurement and reporting principles used by other data and communication systems. They should not inherently have higher failure or error rates. However in some cases these problem rates are higher as demonstrated by monitoring and recording. In other cases bad data quality may be just a perception created by poor data handling. Some of these differences and contributing factors include:

- Phasor data is sent at a high rate, much higher than comparable data systems
    - Data problems become more visible as they occur more frequently
    - The communication provisioning may be insufficient for data requirements
    - Data recording sometimes overruns due to lack of space planning and maintenance
- Phasor data systems may not be fully validated before use
    - The PMU may be improperly or incompletely installed
    - The measurements may not be calibrated
    - The measurement may not be validated against a reference
    - The validation reference may be inaccurate (eg, infrequent SCADA calibration)
- Phasor measurement data display and analysis may make errors more visible to users

    - Dropouts improperly patched or discarded create the impression of bad data reports
    - Flags indicating limits of data use may be ignored (i.e., bad phase angles are displayed)
    - Power flow computed from phasors may show errors but SCADA power flows are usually taken from a state estimator which removes bad readings, so they are not seen
    - Phasors are timetagged at the PMU, so the measurement time can differ from that of SCADA which is timetagged at the master
- Phasors offer new visibility and operation capability
    - Phasors directly report phase angle; using these measurements directly for system state solution creates problems when there are measurement errors. Traditional state estimation rejects bad data, so these errors are not seen
    - Phasors show dynamic activity that SCADA cannot see; this can create the impression phasors are showing noisy or erroneous data
    - Latency for phasor data reporting is an issue since use for real-time control is planned. SCADA was never intended for this use, so latency was never a concern

Most of these problems can be addressed with good design, suitable equipment selection, and thorough validation of the measurements. This can eliminate the 'built-in' problems. It will certainly reduce the number of problems that have been observed. But it will not eliminate problems that occur due to failure during operation. For this, the sub-systems used in a phasor data system have the capability for self-monitoring. The PMU can detect and report internal processing errors. Timing errors can be detected by the timing source and flagged in the data. The communication and data processing elements (such as a PDC) can detect errors and flag them appropriately. Data display and analysis systems can read error and exception indications to suitably process the data.

However no matter how well the design and implementation is carried out, failures and other problems that need to be detected will occur. The only way to assure that data is always valid and accurate is with an on-line validation system. It needs to detect all errors that will adversely affect the applications being served. Ideally errors would be corrected as well, but this is a not always possible with the current technology. The question is what can be done that is reasonably effective at detecting problems yet within the reach of current systems and technology.

Flags and indications imbedded in the data are the easiest indications to work with. For cases not covered with self-checking flags, other detection such as measurement value checks and cross checks between the values from a PMU. More errors can be detected by comparing measurements with the known characteristics of the physical power system.

However some measurement errors, like calibration, are not visible to the measuring and communicating systems. These errors must be detected by comparison with a reference or by closed system examination, such as state estimation using a system model. Either option brings the problem to a higher level of complexity. Redundant measurements provide a measure of reliability but require more measurement and on-line decision making capability. State estimation with bad data detection requires a complex model well as enough measurements to support over-determined equations (more

measurements than unknowns) to detect errors. With rare exception, phasor measurement systems in operation today are not extensive enough to support over-determined state estimation equations.

3.3 EPG approach to the problem

EPG approached the problem based on experience with utilities using phasor systems. Many of the observed problems were due to insufficient design. Others were due to equipment problems, and still others due to operation and maintenance issues. It appeared that if the institutional problems were addressed, most of the observed problems would disappear and the remaining ones would be very manageable. So the first step was to find out how the utilities themselves were dealing with the data quality issues.

The initial stage of the project surveyed existing experience with synchrophasors, focusing on the ARRA projects. Few projects were to the point of full operation, so they were not yet assessing performance. Consequently there was not much information about data problems. The survey did point to a lack of effective installation and validation procedures being consistently used. The next stage of the project was publication of a 'best practices' that presented comprehensive installation and validation procedures as well as planning and design recommendations. If followed, these could reduce or eliminate many of the problems that have been observed.

The next stage was the production of a DV&C prototype utility that could perform validation and conditioning on a real-time data stream. Options ranged from simply checking the imbedded data quality flags to correcting data errors using redundant measurements or a state estimator. Based on observations from working with utilities and the survey in the first part of this project, it was apparent the few utilities could take advantage of the more complex options as they had few measurements compared with the power system. EPG decided to focus on "data-centric" methods that do not require external information such as a full system model.

EPG analyzed data validation from this perspective issue and broke the process down into six stages. Each stage addresses a process in the reception, decoding, and management of data. An algorithm examines the data at each stage to detect data corruption that can be addressed at that point. The algorithms cover the types of errors by the stage of processing. These are:

1. Module 1- Communication Interface: This module is designed to check for errors that may be introduced in the communications chain such as dropped bits, incorrect message frames, and communication system CRC errors.
2. Module 2 – Message Characteristics: This module checks for message format errors such as length, destination address, type identification, and message CRC16.
3. Module 3 – Timestamp: This module checks time tags for sequencing, data rates and transmission delays.
4. Module 4 – Quality Flags: This module utilizes all the flags available in the C37.118 standard to distinguish between good, bad, and uncertain measurements. Bad data is converted to NaN, suspect data is flagged, and all data is passed on to the next module for further processing.

5. Module 5 – Data Characteristics and Self-Checking: This module incorporates algorithms to check for unreasonably high or low values of voltage, current and frequency, data that is stale (not refreshing), and excessively noisy. Depending on severity, data that fails testing is declared bad and set to NaN or uncertain and flagged.
6. Module 6 – Topology Checking: The last module uses system topology to build algorithmic logic checking. For example, the sum of currents into a bus should be 0, and voltages at the same bus should be the same.

The modules are linked together into the data validation application as shown in Figure 2. Each one treats the data at an appropriate stage, such as the communication error detection at the communication interface stage. A data quality flag is assigned to each data item at the first stage and carried with the data. The PMU data flags apply to the whole data frame, but these quality flags apply to each data value. Each phasor has a flag for magnitude and a flag for angle. Each frequency and ROCOF measurement has a flag. Flag generation and contents are detailed in the following sections.

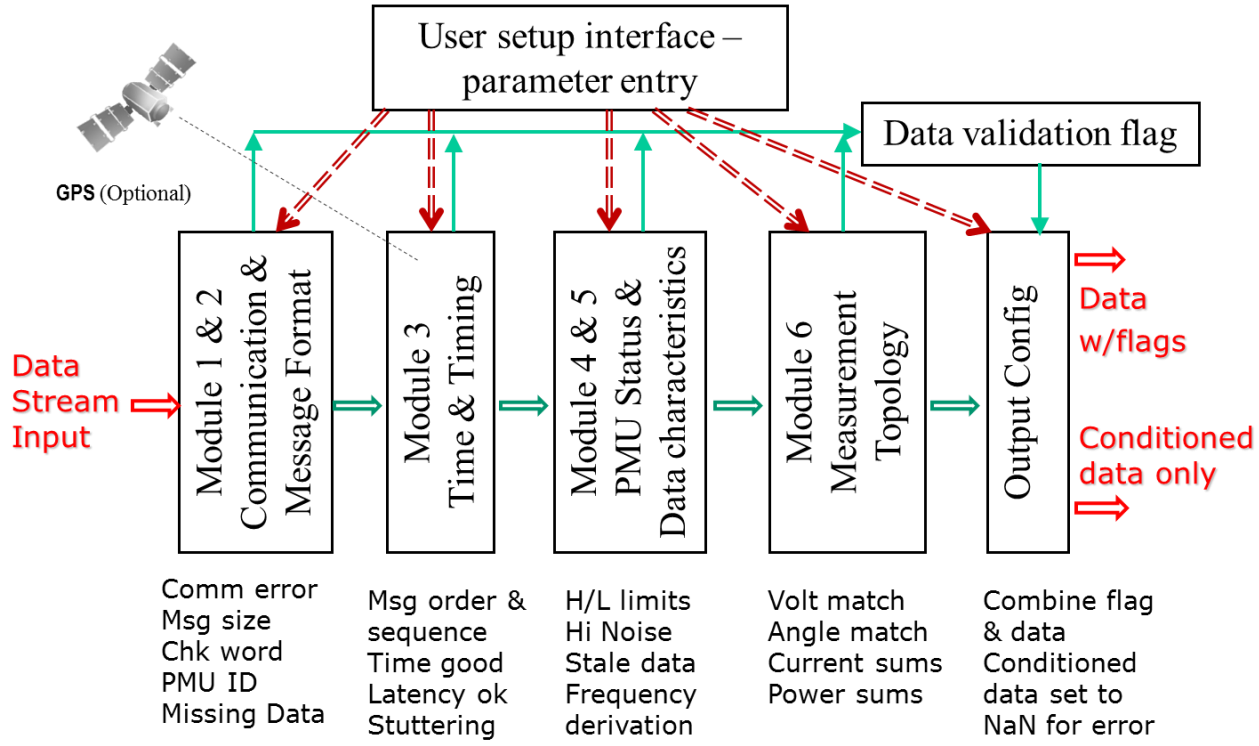

Figure 2. Structure of Data Validation Application

After coding and linking the modules together, EPG tested them with samples of good data and then data with errors of the type that the prototype was designed to detect. After the usual bug detection and fixes, the prototype was ready for demonstration and evaluation, followed by improvements if some were identified.

3.4 DV&C requirements

The overall requirement is to detect data and resolve errors that could cause failure in analysis or other functions that use the data. This is a very general approach and not very useful as a guideline, since errors that could cause failure of one application may be ignored by another application. The contract specifies that the algorithm detect data errors using a variety of methods such as simple error checking and flags and include more complex approaches such as topology, model comparisons, or SCADA comparisons. It specifically includes the ability to detect:

- Loss of data from one or several PMUs
- Loss of signals in a PMU
- Stale (non-refreshing) data
- Inconsistent data, data rates and latencies
- Off-sets in signal magnitude and phase
- Corrupted and drifting signals in a PMU
- Corrupted and drifting time reference in one or several PMUs
- A combination of several issue described above

EPG has taken this basic set of requirements and added capability to detect:

- Communication errors
- Message corruption of any type
- Delays in signals
- Timetag errors
- Noisy signals

The algorithm should operate in real time with minimal delay so it can be used with most if not all applications. There should be a record of errors detected and any modifications of the data.

3.5 Terminology and definitions

For the purposes of this project, EPG uses data terminology to indicate specific conditions. These are defined in the following table. This terminology has been used in studies and may in those contexts indicate different criteria, so these definitions are specific to this project. It is hoped that in the future we will have a general adoption of specific definitions so the terminology may be consistently used in all of this kind of work.

Term	**Definition**
Raw data or raw measurements	Data as sent from the PMU before any DV&C is applied
Validated data	Data that has passed validation checks, but not necessarily accurate. Here valid means the data is correctly applied to the measurement and is not corrupted. It may or may not pass

	reasonableness checks.
Fully validated data	Validated data as above but also passes reasonableness checks (outliers) and topology checks as can be applied. This data will be acceptable (no uncertainty discovered) or will be set to NaN or a specific user defined value.
Conditioned data	Data that has been fully validated and compared against a model or other means to assure reasonable accuracy. This data will have that passed all tests within the limits of comparison or will be set to NaN.
Fully conditioned data	Conditioned data that also has been modified to an improved value if the accuracy comparison indicates an improvement with a high degree of certainty. "Conditioned data" will only set bad values to NaN and does not otherwise modify the measured value.

4. DV&C Algorithm

4.1 Overview

As described previously, the prototype validation and conditioning application was implemented in software as a series of algorithms that examine the data. For purposes of description, this prototype will be simply called the "DVC" as shorthand for the data validation and conditioning prototype.

The overall DVC system diagram is shown in figure 3. The inputs are user data and configuration and measurements from the system. The user entered data includes:

- Data input configuration. The DVC needs communication parameters to connect with the measurement system through a C37.118 data link. This includes both the measurement and configuration data.
- Data comparison settings. The user must specify limits for various data validation comparisons. These limits can be tailored to be close to expected operational parameters such as expected power flow or more general like limits for operational voltage or current. The user also has to specify which errors will be used to declare data unusable or simply questionable.
- Data output configuration. There are several choices for how data will be treated at the output.

The entries from the data system are configuration and measured data. Configuration information tells the DVC the data message contents which is necessary for parsing the data from the messages. Configuration also includes the data identification (name, type, format, etc) and relevant scale factors.

Configuration has to be available any time the data message changes. The measured data is sent as a continuous stream of timetagged data frames.

Outputs from the system are data, data flags, and monitoring information. The system message output includes counts of errors of various types and general operation reporting so the user can see if there are problems as well as look at historical error accumulation. This output in the prototype is implemented with both a GUI and a text record.

The data outputs can include raw (unconditioned) data or conditioned data, each with or without the quality flags. Since the output is C37.118 compliant, the DVC will also output the appropriate configuration.

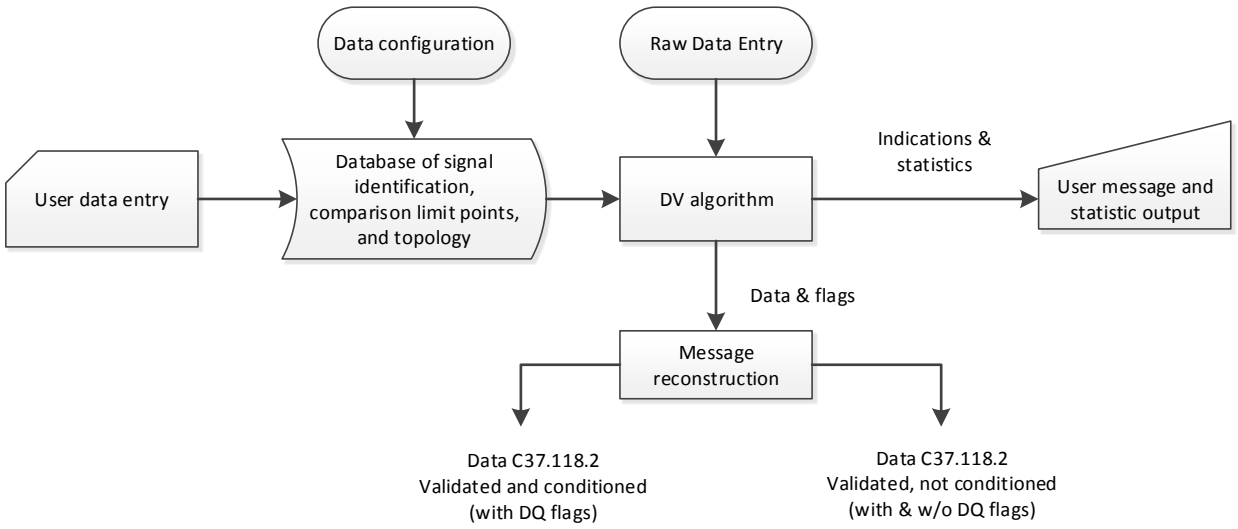

Figure 3. Overall DVC diagram

The user entry system, configuration database, and user output system are specific to the DVC software implementation and the type of hardware it will run on so are not described in this document. The required user entries, validation processes, and algorithm generated information are fully detailed.

4.2 Measurement Validation

A PMU creates and transmits synchrophasor data. The data from a PMU using the C37.118 protocol is sent as timetagged frames, each of which includes the following information:

- Timetag. This includes second-of-century and fraction-of-second fields that set the time of measurement to within 1 microsecond.
- Status. This is a 16-bit digital indication that includes quality information for all the data in the entire frame.
- Phasors. Synchrophasor estimates for one or more voltage or current values.
- Frequency. Estimate of the local power system frequency.

- ROCOF. Estimate of the local rate-of-change-of-frequency.
- Digitals. Boolean number representation for status indications, packed as 16-bit numbers.
- Analogs. Digital value for a sampled analog signal.

The DVC algorithms only examine phasor, frequency and ROCOF data. C37.118 data includes digital indication values (Boolean) which only have user specified assignments so there are no standard values to assess. It also includes analog values which are digital samples of analog valued signals such as a power measurement, a controller setting, or a temperature measurement. Since these are also non-standard and user specified, they are not assessed by this algorithm. In the future if there are specific value assignments, they can be included in quality assessment.

With data from a single PMU, the timetag and status apply to all values in the data frame. Any errors or data quality degradation indicated in these values need to be considered when processing all the data in the frame.

Data from many PMUs may be combined into a larger data set by a PDC or other device. When that combined data is sent using C37.118, there is one timetag for all data in the frame, but the individual status for the data for each PMU is retained. It may be modified to indicate further processing or modification by the PDC. The status still applies to all the data from that PMU.

Data from each PMU may contain several phasor values in addition to one frequency and one ROCOF. Each one of these measurements may have errors that need to be detected and flagged or corrected. Furthermore, phasor values include both magnitude and phase angle information. There are conditions that may cause an error in one and not the other. Some uses for phasors also allow using one component and ignoring the other. Consequently phasors need to be in a format where magnitude and phase angle errors can be assessed individually. Frequency and ROCOF also need to be assessed individually.

Phasor, frequency, and ROCOF can be sent in 16-bit integer or floating-point (FP) format. Further, phasors can be in rectangular or polar coordinates. There are some historical and logical reasons for using these different formats, but that won't be explored here. For processing, all phasors will be converted into FP polar and frequency and ROCOF into FP. The reasons and justification for doing this are:

- Polar coordinates allow separating and conditioning magnitude and phase angle individually
- FP has more resolution than integer and there is no loss of resolution if the measured value is converted from integer to FP. The reverse may not be true
- All numbers can be converted to line side engineering quantities, so no extra scale factors are needed
- All comparisons can be done in standard engineering values
- Invalid data can be represented unambiguously in FP using not-a-number (NaN)
- Conversion from integer to FP and rectangular to polar coordinates are well known and assumed to be error free

Once data is parsed from the message by PMU and then into individual measurement quantities, each measurement value can be examined using relevant criterion. Each measurement may have several stages of examination. At each stage, the measurement value may be determined to be in error, questionable, good, or unknown. A quality flag for each data item is needed to carry the determination through the examination process. The measurement inherits an initial quality from the PMU status, which applies to all measurements from a particular PMU. The flag is updated as needed and then is available as an output indicator of the overall data quality. Parsing data and setting up quality flags are illustrated in figure 4 below.

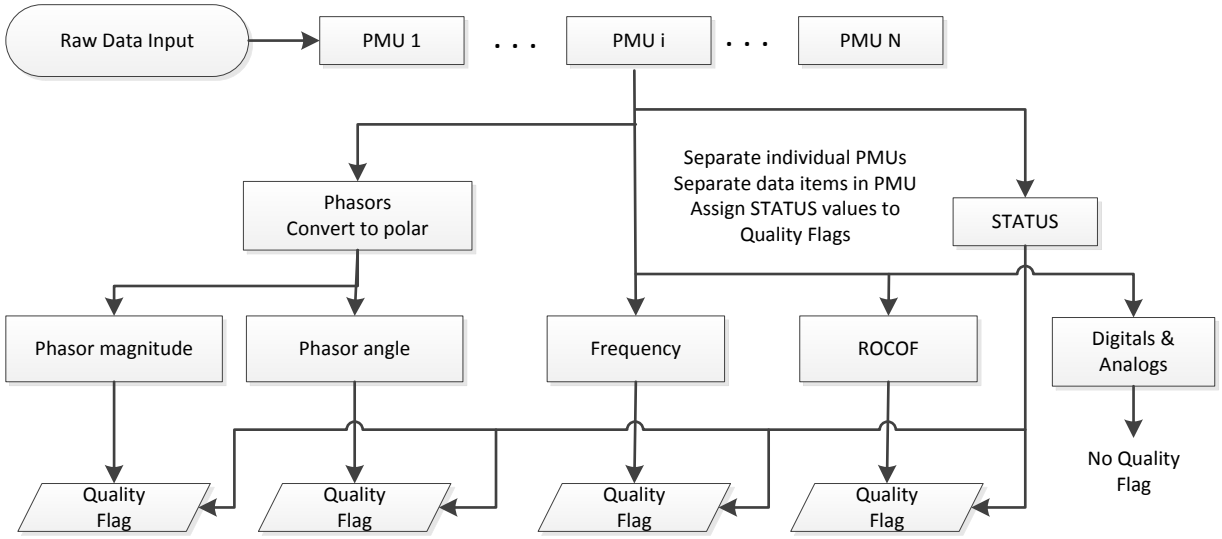

Figure 4. Data separated into components and assignment of Quality Flags

The quality flags are 8-bit flags that encode the measurement quality as determined by the algorithms. They have three fields that indicate (1) overall quality, (2) sub-status, and (3) limits. Quality determined by the algorithms is mapped into these indications as described in the following sections. The quality flags may be used internally to aid in processing and may be included in the data output. They can map as 16-bit quantities in pairs: for each phasor there is a magnitude and an angle flag, and for frequency a frequency and a ROCOF flag. C37.118 outputs can include 16-bit digitals or 16-bit analog outputs, either of which matches these quality flag pairs. Details of the flags are described in Appendix A.

4.3 DVC processing and algorithms

As described in the previous sections, data is input to the DVC as a C37.118 data stream. This requires a communication interface and supporting features. Since this is the first place the DVC sees the data, the first algorithm uses error checking provided by communication interface. Five more algorithms follow, each examining further characteristics of the data. The algorithms are designed to successively build on each other to simplify and strengthen the determination. For example, corrupted data as indicated by a check word error should not be used at all even though it could be any value, even one that seems good. Checking the CRC and marking bad data at an early stage avoids further checks and eliminates the possibility it may pass through as good data.

The DVC's internal data buffer initializes frequency, voltage, and current for every sample as NaN with quality code as NOT CONNECT (00001000). If a sample has the correct CRC and can be parsed, the above NaN value and quality code will be overridden by parsed values. If a sample is not received, its values will be kept as NaN with quality code NOT CONNECT (00001000) to indicate data is missing. If a sample is received but the CRC is wrong, its value will be kept as NaN and the quality code will be set to COMMUNICATION FAILURE (00011000).

Figure 5 illustrates the overall operation of the DVC and the component algorithms. The raw data comes in on the left and the flags that are assigned to each measurement on the right. Data comes in as frames since the first tests deal with the entire data frames. Errors at those stages have to be marked in all the quality flags for the entire frame. Each algorithm performs tests and updates the quality flags as required. Note that the first 3 algorithms test the entire frame of data, so data is not parsed until entering the 4 module. The 4th module looks at the status, but also has to mark individual data flags. The 5th module has to look at the individual measurement, so has to convert measurement values to polar, FP, and apply all required scaling. The operation of each module is detailed in the following sections.

Overall Algorithm for Data Validation

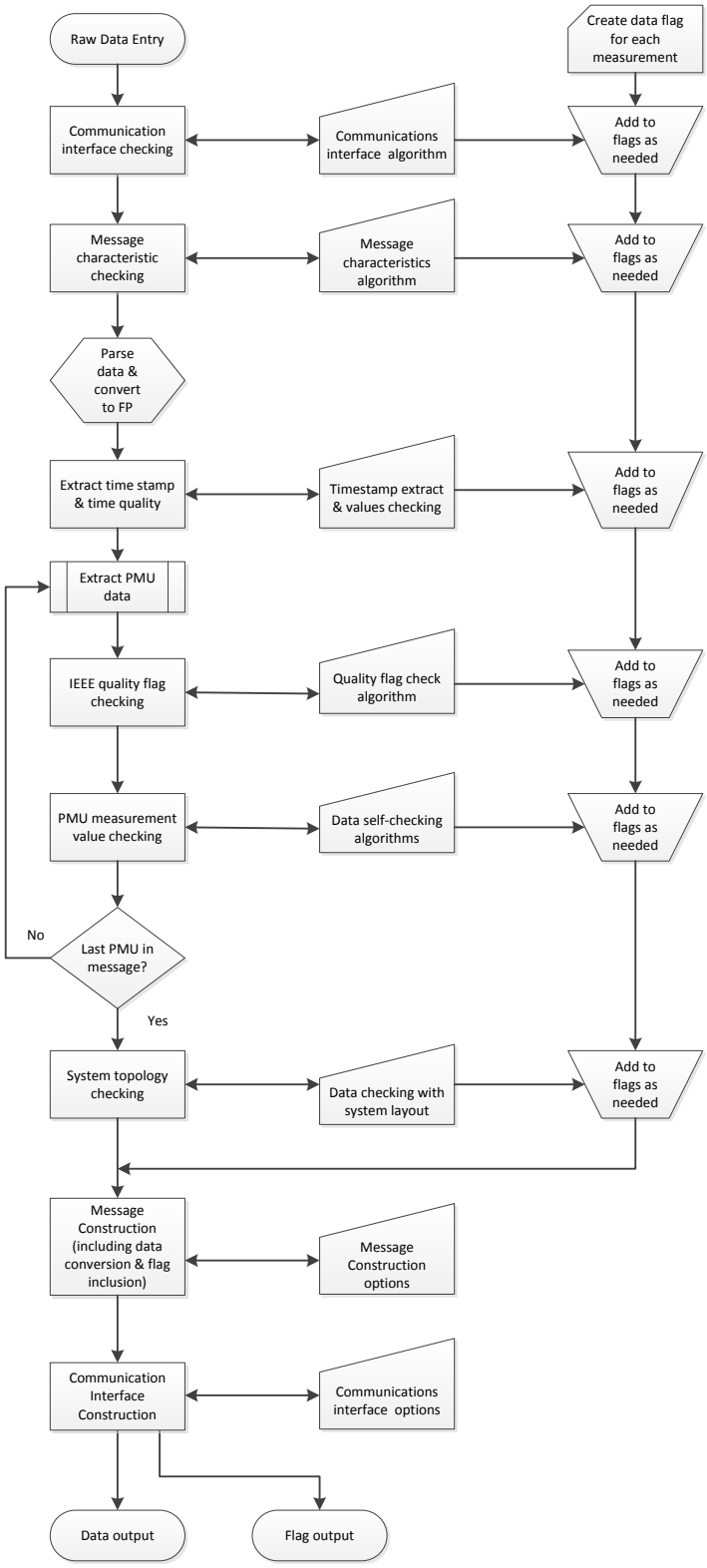

Figure 5. Overall DVC algorithm organization

4.4 Communication Input Processing Module

The first module (Figure 6) processes any information received from the communication interface. This will indicate errors such as framing and Ethernet CRC errors, if reported. Usually the interface will discard the data and no errors will be presented. In that case, the algorithm will simply indicate lost data. If the interface does report an error it most likely will be a fatal error which is considered as bad data. If data is also reported, it should be converted to NaN to prevent any use as good data.

Communication interface checking

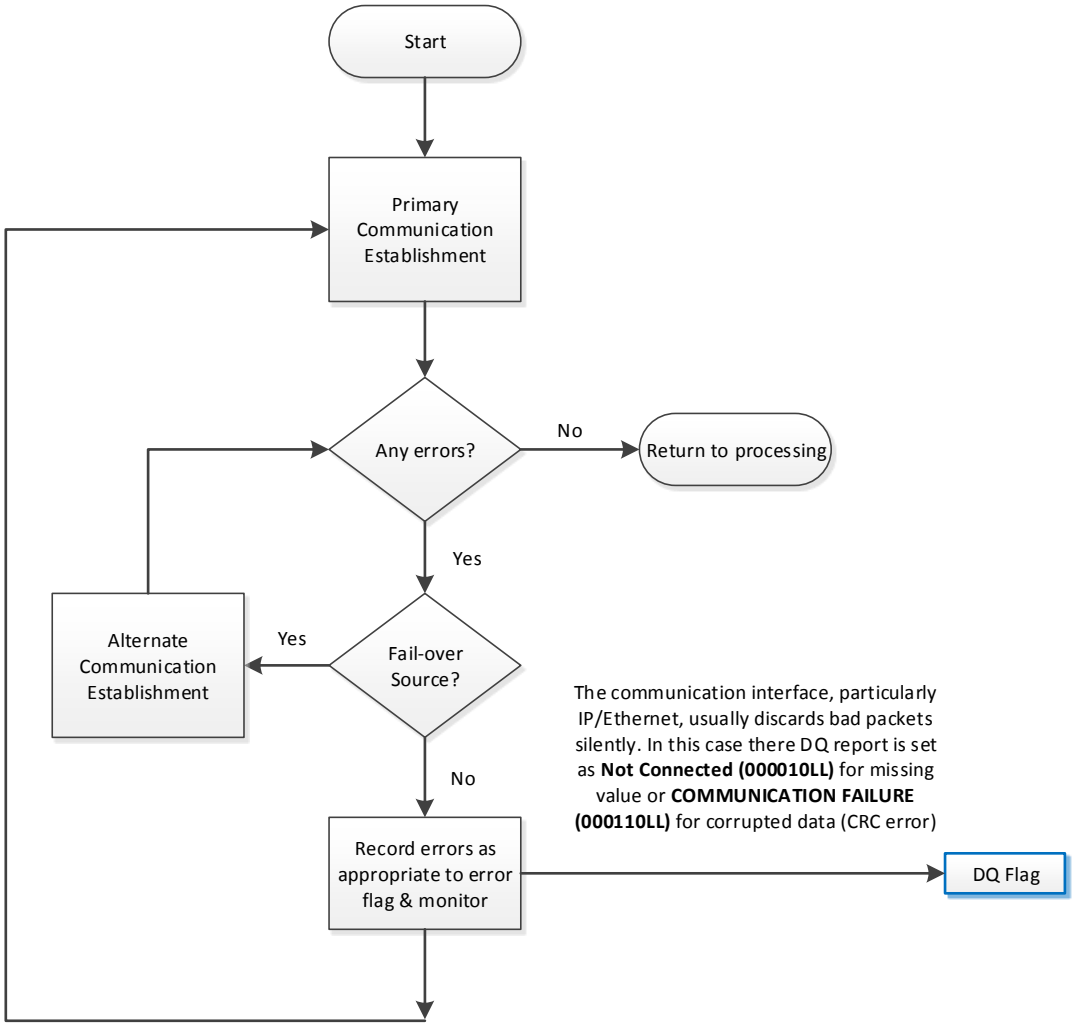

Figure 6. Communication Interface Processing Module

Table 1. Communication Interface Processing Outputs & User Entry

Test	Outcome	Flag Indication	Data Conditioning	User Entry
Error detection by interface (all protocol layers below application)	Data not reported (interface discards bad data)	Bad Data, Not Connected (00001000)	NaN	None
	Corrupted data reported at Ethernet level	Bad Data, Not Connected (00001000)	NaN	None

4.5 Message Characteristics Processing Module

The message characteristics module (Figure 7) examines message characteristics which could indicate a corruption or intrusion. These are the sync word, message length, message CRC, and intended destination. If there is an error up to this point, it is considered fatal so the data is set to NaN to prevent use and the QF is set to bad. In this case, no further checking is necessary.

If a sample has the correct synch word, message length, CRC, and intended destination, it will be parsed. After parsing successfully, the parsed values will be initially marked as GOOD for later processing. Otherwise, it a sample is received but cannot be parsed or CRC error, its values will be kept as NaN and the quality code will be set to COMMUNICATION FAILURE (00011000).

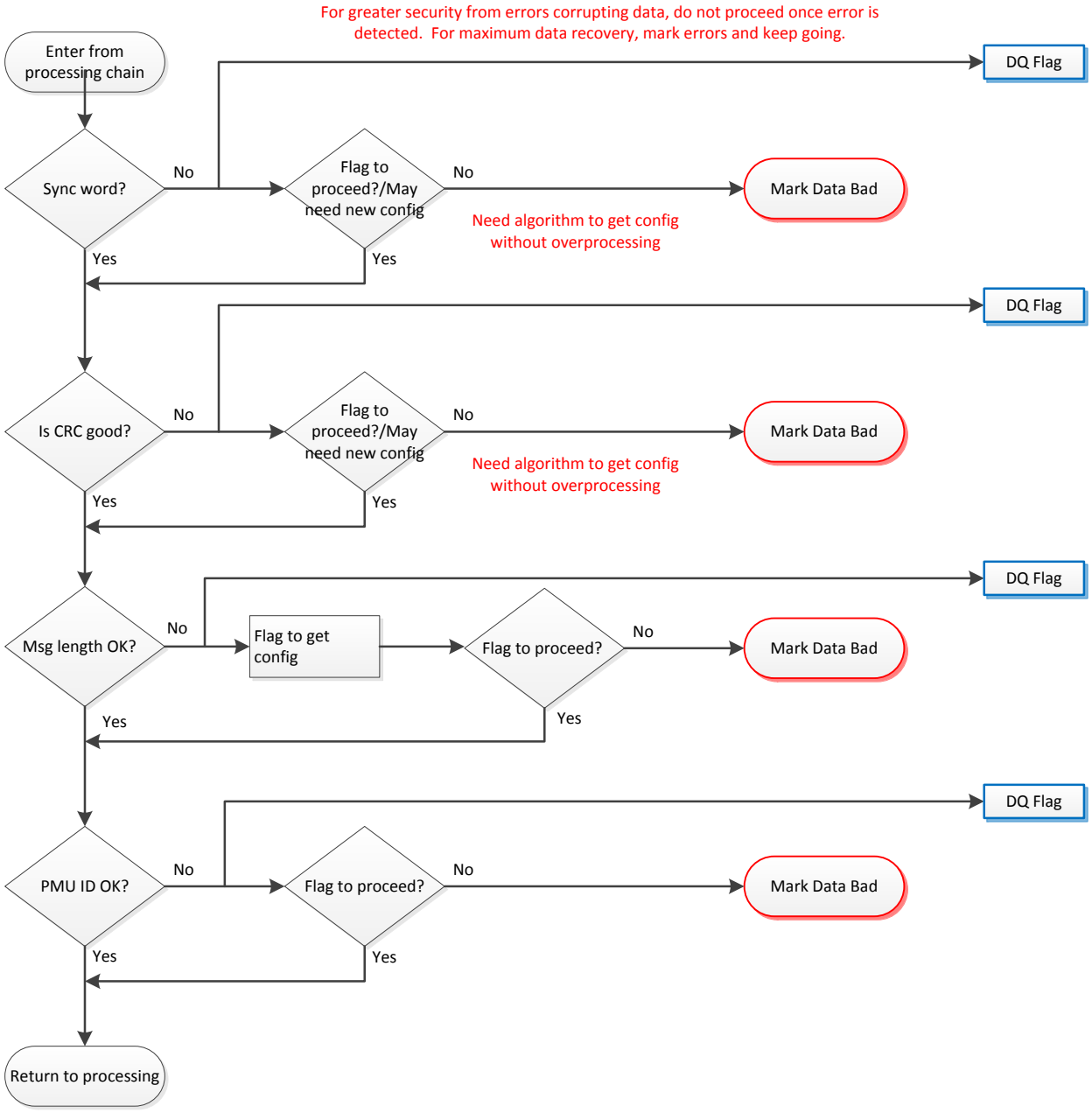

Figure 7. Message Characteristics Processing Module

Table 2. Message Characteristics Processing Outputs & User Entry

Test	Outcome	Flag Indication	Data Conditioning	User Entry
Synchronizing word correct?	Incorrect sync byte, message	Bad data, configuration error – wrong format	NaN	None

	number, version number	(00000101)		
Message CRC good?	CRC bad	Bad data, CRC error (00100100)	NaN	None
Message length good?	Message length number and actual length do not match	Bad data, configuration error - message length (00000110)	NaN	None
PMU ID correct?	PMU ID indicates a different destination	Bad data, message wrong place, configuration error (00000111)	NaN	None

4.6 Message Timetag Processing Module

The timetag module (figure 8) extracts the timetag and examines it for errors. It checks to be sure the timetags are in sequence and none are missing based on the programmed data rate. If the timetags are in correct order it calculates latency based on arrival time. If accurate time is not supplied to the algorithm, this function can be skipped. It records any errors in the quality flag.

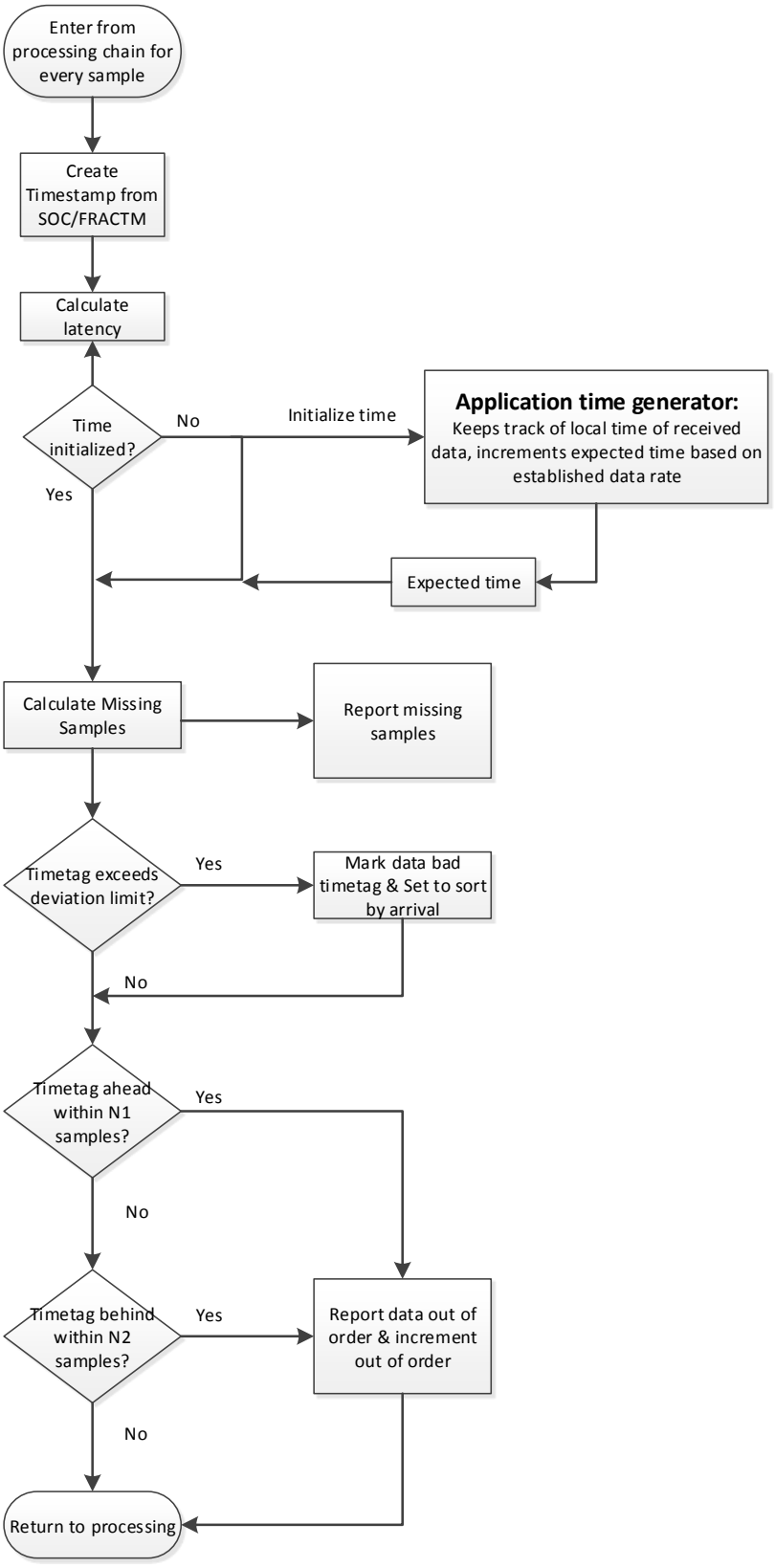

Figure 8. Message Timetag Processing Module

Table 3. Message Timetag Processing Outputs & User Entry

Test	Outcome	Flag Indication	Data Conditioning	User Entry
Time stamp exceeds deviation limit?	Yes	Bad data, Sensor Failure (00010000)	Set PMU Status word to sort by arrival	Deviation Limit
Time stamp too far ahead?	Time stamp is ahead of time reference (system time)	None	None	Limit for samples ahead of current time
Time stamp too late?	Time stamp is behind of time reference (system time)	None	None	Limit for samples behind current time
Missing Sample?	Timestamp not found	None	None	None

4.7 Data Conversion and Scaling

Referring to the overall diagram (Figure 5), after the timetag is processed, the individual data values need to be extracted from the overall message. The message is comprised of blocks of measurements from each PMU. Each PMU block starts with a status that provides data quality for all the data in the message. Following that is phasor, frequency, analog, and digital information respectively. The processing system needs to first extract all the signal information for each PMU from the message. Then it needs to apply scaling for all data as provided by requirements (frequency & ROCOF have fixed scaling) or from the configuration file. Phasors, frequency, and ROCOF should be converted to floating point polar (phasors) format at this point also. From here on, phasors, frequency, and ROCOF can be evaluated as line scaled, engineering quantities.

4.8 PMU Status Flag Processing Module

The PMU status module (Figure 9) looks at the C37.118 status flag for each individual PMU. Since these flags cover the whole PMU, which includes several measurements, these check results have to be applied to all the quality flags for each measurement from that PMU. Since the message may contain more than 1 PMU block, the algorithm cycles through this module for each PMU block (see Figure 5). This module checks for missing data, PMU errors, data out of sync, test mode data, and data with a local timetag.

The PMU status module in Figure 9 shows the processing logic. Processing proceeds through the flag bit tests which are based on the meanings indicated in the standard. Some flag settings always indicate the data is bad and others only indicate a degree of impairment. The user may prefer to elevate certain indications to be a fatal error; User Flags indicate where the analysis skips to indicate a fatal flaw based on these user preferences. Otherwise the processing proceeds according to the standard implications.

In all cases, the result of testing is shown in the Data Quality flag. It first examines the PMU time quality (PTQ). A PTQ > 4 indicates a possible error > 100 μs which is below an accuracy that should be used, so the sync error bit 13 is set. PTQ was established with the 2011 standard. In the previous 2005 version, these bits are set to 0, so they will not interfere with this test.

Next if bits 15 & 14 are cleared, then the data is valid and there are no PMU errors, so examination can proceed to checking synchronization.

Conversely missing data is usually indicated by setting all 4 bits 15-12. Since this is not by standard but by convention, the user can set flag that indicates 4 bits set means the data was missing and the filler data should be set to bad. If the user does not set the flag, the data is left as is for further processing (it could be usable data).

Bits 15-14 = 01 indicates that a PMU error has been detected by the PMU. The nature of the error is up to the PMU vendor to publish. If the user is advised that this is a fatal error, the user can set the flag to discard the data (set the data to bad). If the user wishes to further process the data, the flag can also be set to continue.

Bits 15-14 = 10 may indicate the data is invalid or may indicate the PMU has been set to test mode. Again the user may set to discard data with this indication or ignore it depending on what the flag is most likely to indicate.

Bits 15-14 = 11 always indicates the data is invalid so the data is always set to bad (discarded).

Bit 13 = 1 indicates the PMU is not in sync with the UTC time source, so the measured phase angle is suspect. In some cases, the user may want the phase angle changed to indicate it is bad data at this point so it is not available to any further calculations. The user set flag will cause the data set to bad at this point. Otherwise, the flag can be used by further modules to make the decision for whether to use the angle or not. Note that measurements form the same PMU will be in sync with each other even if GPS sync is lost, so power calculations using phase angle from the same PMU are valid. The user flag allows flexibility in this decision.

Bit 12 = 1 indicates the PMU timetag was applied by a remote device as a best guess to replace a bad timetag. Since it is not a precisely synchronized time with the UTC time source, the measured phase angle is not usable. As with bit 13, there may be cases where the user wants the phase angle set to bad data and other cases where they do not. There is a user settable flag to make this choice.

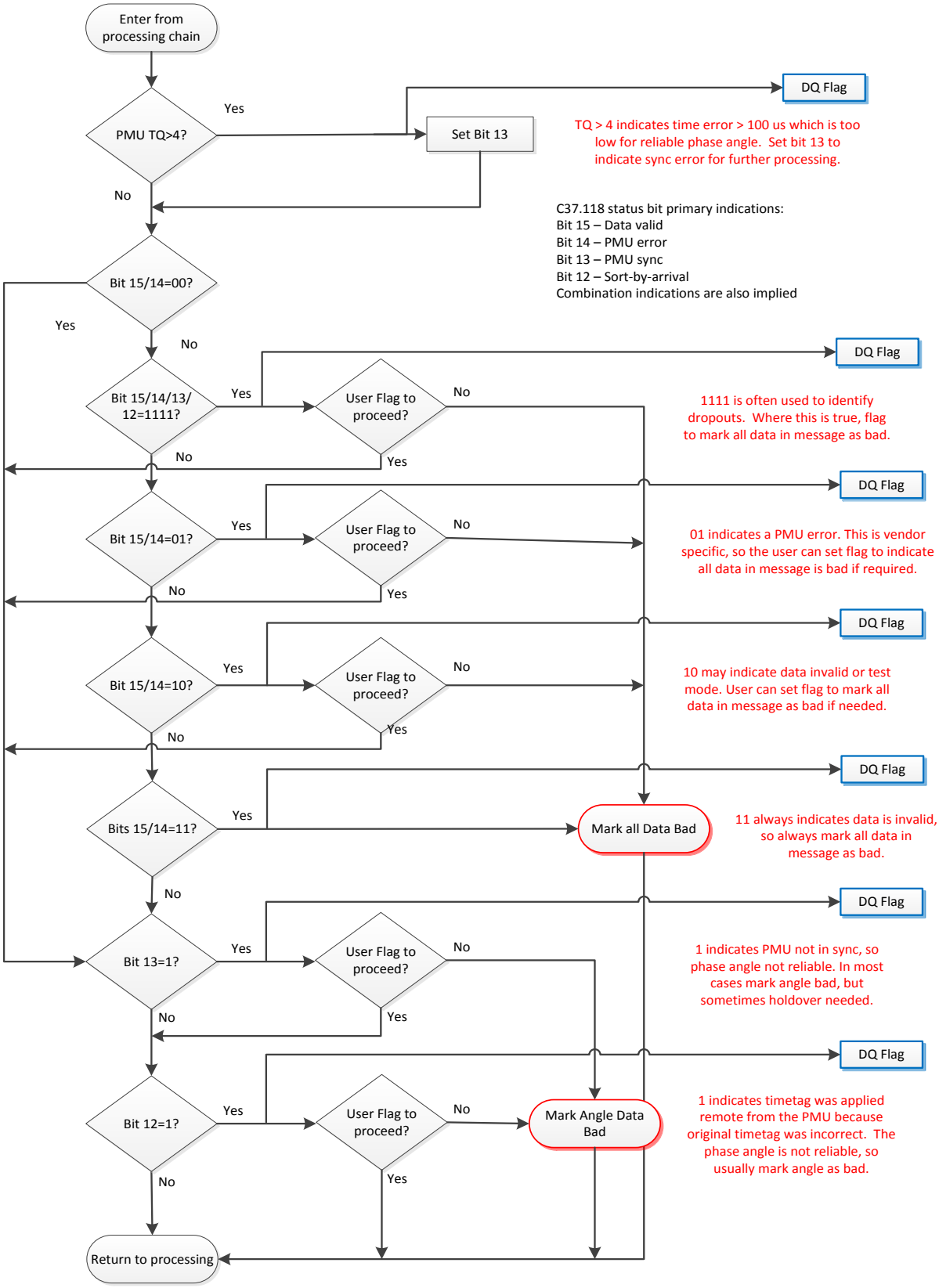

Figure 9. PMU Status Flag Processing Module

Table 4. PMU Status Flag Processing Outputs & User Entry

Test	Outcome	Flag Indication	Data Condition	User Entry
Time quality at usable accuracy for synchrophasors?	TQ>4	Uncertain data, sensor not accurate (01010000 0)	Sync bit 13 set to 1	Enable or disable TQ validation
PMU block & PMU good?	Bits 15/14 =00	None – flag clear (00000000)	Data ok	Enable or disable PMU Data valid checking
PMU status all bad?	Bits 15-12 =1111	Bad data, dropout (000010LL)	All data set to NaN in output when condition is enabled	User flag set to indicate condition is dropout
		Bad data, unknown (000010LL)	Data left as received	User flag set to leave data for further processing
PMU error or test mode?	Bits 15/14 =01	PMU error, do not use data (01000000)	Data set to NaN in output when condition is enabled	User flag set to indicate condition is PMU error that is considered fatal so data is bad
		PMU error, use data with caution (01000000)	Data left as received	User flag set to indicate condition is likely non-fatal PMU error
Data valid flag?	Bits 15/14 =10	Data invalid, do not use data-cause out of service (00011100)	Data set to NaN in output when condition is enabled	User flag set to indicate condition is data is invalid, not usable
		Test mode, do not use data (00011100)	Data left as received	User flag set to indicate condition is test mode

Valid & PMU error flag?	Bits 15/14 =11	Data invalid, do not use data-cause unknown (00000000)	Data set to NaN in output when condition is enabled	None
PMU sync flag	Bit 13 =1	Time sync bad, phase angle not usable, sensor not accurate (01010001)	Phase angle data set to NaN in output when condition is enabled	User flag set to indicate setting all phase angle data to invalid to prevent use
		Time sync bad, use phase angle with caution (01010001)	Data left as received	User flag set to leave phase angle data for use with caution
PMU sort-by-arrival (local timetag)	Bit 12 =1	Time sync bad, phase angle not usable (01100010)	Phase angle data set to NaN in output when condition is enabled	User flag set to indicate setting all phase angle data to invalid to prevent use
		Time sync bad, use phase angle with caution (01100010)	Data left as received	User flag set to leave phase angle data for use with caution

4.9 Data Self-Checking Module

The data self-checking module (figure 10) compares individual data items with expected characteristics for each type of data such as reasonable value ranges, limited signal noise, changing measurements, and derived signals that match. Signals lacking these qualities are suspected bad or may be declared definitely bad. These characteristics are based on the data itself rather than relations with other measurements. All incoming data should be scaled and in floating polar format. The processing in this module should be followed with each measured quantity.

The first test looks at high and low limits for voltage and current magnitude. Voltage is usually restricted within a narrow range, allowing the use of low and high limits. Current can range from 0 to a very high value though some links are limited. In all cases, transient phenomena such as faults can cause momentary spikes past limits that should be ignored. This test needs a counter that will allow a few sample values up to 1 s of values to exceed the limit without declaring a data error. This could be a user set delay, but here is fixed as it does not usually need to be individually tailored.

The second test looks at high/low frequency limits. Frequency is usually derived from a voltage signal; if this signal does not have high enough amplitude for a good frequency measurement to be made, the frequency should be declared bad. As with voltage and current magnitude, transients can cause momentary spikes, so this test needs a counter to be sure the limited is exceeded long enough to indicate a bad measurement.

The next test is for excess noise on the signal. The signal will have some variation and some noise, but the variation will be within the measurement frequency passband and the amplitude will be much smaller than the signal value itself. So what we would observe as a "noisy signal" will be at a higher frequency and with more amplitude that we would expect. The highest frequency that the signal can contain is one just below the Nyquist rate for the give sample rate. So this test is performed by running the signal thought a high pass filter whose cutoff is set near the Nyquist rate for the given data rate. The user can set the threshold for detection of noise. The definition of "excess noise" is subjective, so the setting needs to be set to the user's level of concern. A lower threshold will pick up small amounts of noise; a higher threshold will give fewer noise alarms. This can also be implemented as a signal to noise ratio (SNR) threshold by dividing by the base magnitude of the signal. For production systems using scaled values SNR is the preferred method of implementation.

The next test shown in the module is the stale data test. Stale data is defined as data that is unchanging. This could indicate a the measurement device has a hardware failure that keeps sending the same value or a data collection device is inserting the same value as a filler (and not marking it as repeated data). While formally stale data will be exactly the same value repeated, it may vary just slightly due to conversions between integer and floating point. The stale data test looks to see if any signal has stayed within a very narrow band over a given period of time. If so, it is marked suspected stale. The range and period of time the data must be within are specified by the user.

The last test compares the frequency reported by a PMU with the change in phasor angle in the voltage phasors. Most PMUs compute frequency from rate of change of voltage phasor angles, so these values should match consistently. Frequency computed from phasor angles can be the difference between two angles divided by the time interval between them or using an interval of measurements with a least squares fit to minimize noise. This is then compared with the frequency offset $\Delta f - f_{report} - f_0$. The user must supply a tolerance for comparison and a computation interval length if least-squares is used.[1]

[1] Measurement method contributed by and detailed in NE-ISO report "PMU Data Validation at ISO New England", by Q. Zhang, X. Luo, D. Bertagnolli, S. Maslennikov, B. Nubile

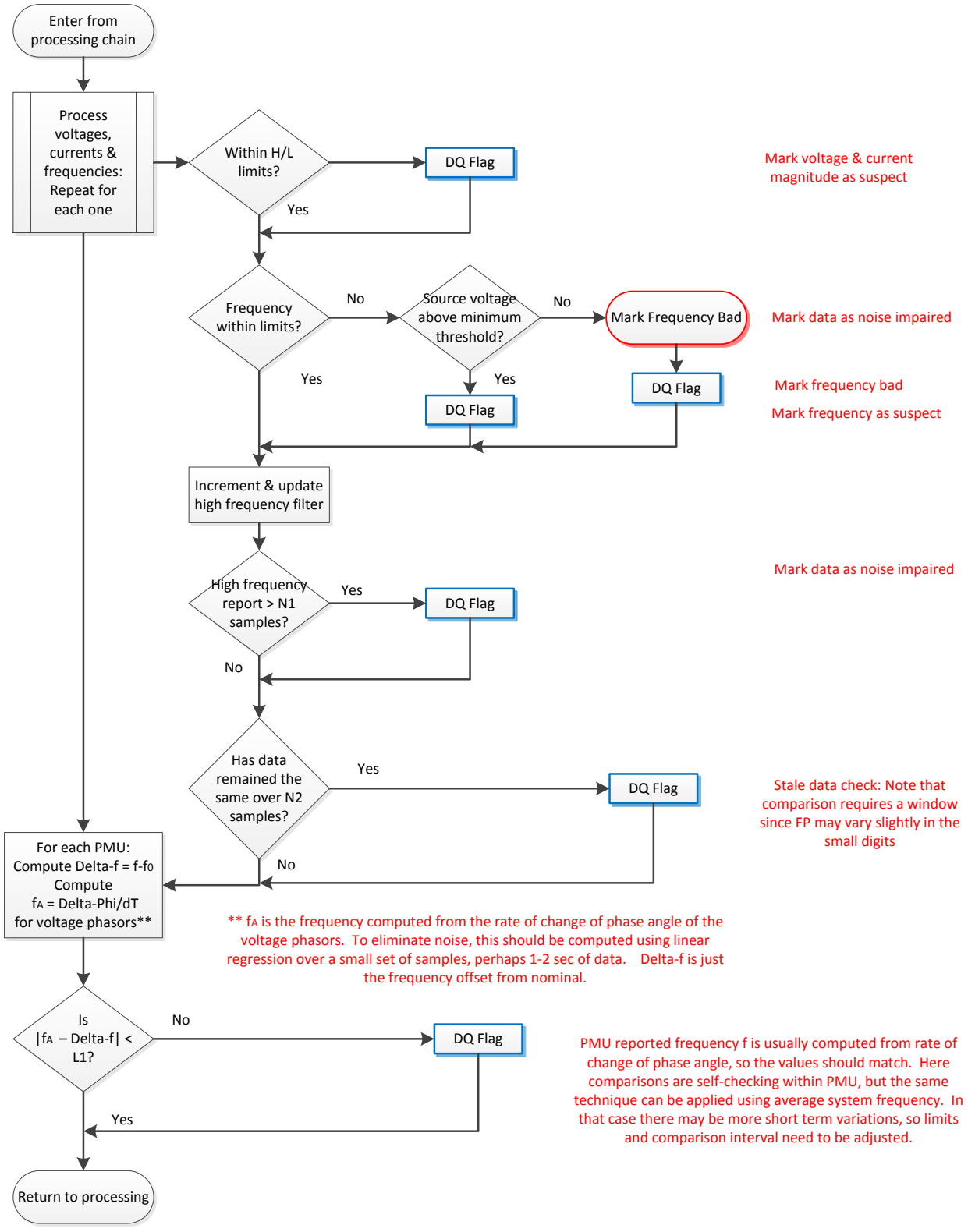

Figure 10. Data Self-checking Module

Table 5. Data Self-checking Outputs & User Entry

Test	Outcome	Flag Indication	Data Condition	User Entry
Voltage and current within high and low limits?	|value| > H_limit	Uncertain data, exceed high limit (01010110)	Data left as received	High limit (H_limit) for each Vmag & Imag
	|value| < L_limit	Uncertain data, exceed low limit (01010101)	Data left as received	Low limit (L_limit) for each Vmag & Imag
Frequency within high and low limits AND source voltage above minimum threshold?	|frequency| > H_limit |source| > threshold	Uncertain data, exceed high limit (01010110)	Data left as received	High limit (H_limit) for each frequency Low limit (L_limit) for each frequency Threshold (threshold) for source signal
	|frequency| < L_limit |source| > threshold	Uncertain data, exceed low limit (01010101)	Data left as received	
	Limit exceeded and |source| < threshold	Data invalid, do not use data-input fail (00010000)	Data left as received	
High frequency content of signal exceed threshold?	|value| > limit	Uncertain data, exceed noise limit (01011100)	Data left as received	Noise threshold limit for each signal
Is signal data stale (not changing)?	|value| < limit for time > minTime	Uncertain data, repeated samples (stale) (01000100)	Data left as received	Band of signal range (limit) that the signal must exceed within given time (minTime)

Is frequency computed from rate of change of phase angles the same as reported Δ-frequency?	Is $\|f_A - \Delta f\| < L1$? $f_A = \Delta\varphi$ where φ is the phasor angle and $\Delta f = f - f_0$ where f is the reported frequency	Uncertain voltage angle or frequency, sensor not accurate (01010000)	Data left as received	Allowable comparison margin L1 and length of comparison window of n samples

4.10 System Topology Comparison Module

The last module is not fully detailed because it is user created and specified. This module allows the user to make comparisons among signals based on the specific topology of the measurements within their power system. For example, if all the currents into one bus are measured, the user can create a sum of these currents and a threshold such that the sum should be nearly 0 (Kirchhoff's law). Another example is the current flow into one end of a line should be close the flow on another end of the line. Both the bus and line comparisons could be done using current instead of power, but this will add the uncertainty of the voltage measurement which may have its own problems. Figure 11 illustrates some possible uses of this module. These examples are not explained in detail since this is not really an implemented part of the algorithm.

The user can write a formula and specify the DQ flag output for failure to meet the threshold. The user can use any of the mathematical, comparison, and logic found in table 6. The user writes the formula using the signal names and symbols in the validation panel. For example, consider the formula:

abs($PMU1.V21.VM$ * $PMU1.C21.IM$ * cos($PMU1.V21.VA$ - $PMU1.C21.IA$) - $PMU2.V31.VM$*$PMU2.C31.IM$*cos($PMU2.V31.VA$-$PMU2.C31.IA$)) < 1

It calculates power measured on line 21 by PMU1 and on line 31 by PMU2 and makes sure the difference is less than one. These are actually two ends of the same line and checking that they match.

A user specified comparison is very flexible and can be used with a wide variety of comparisons. It can also be used on an ad-hoc basis where there is a problem that needs to be tracked down. If some data checking becomes a nuisance, it can simply be deleted.

Table 6. Measurement Topology Checking

Test	Outcome	Flag Indication	Data Conditioning	User Entry
Does the comparison test evaluate to true?	No	Uncertain – topology error (011001LL)	Data left as received	Comparison formula and limit test

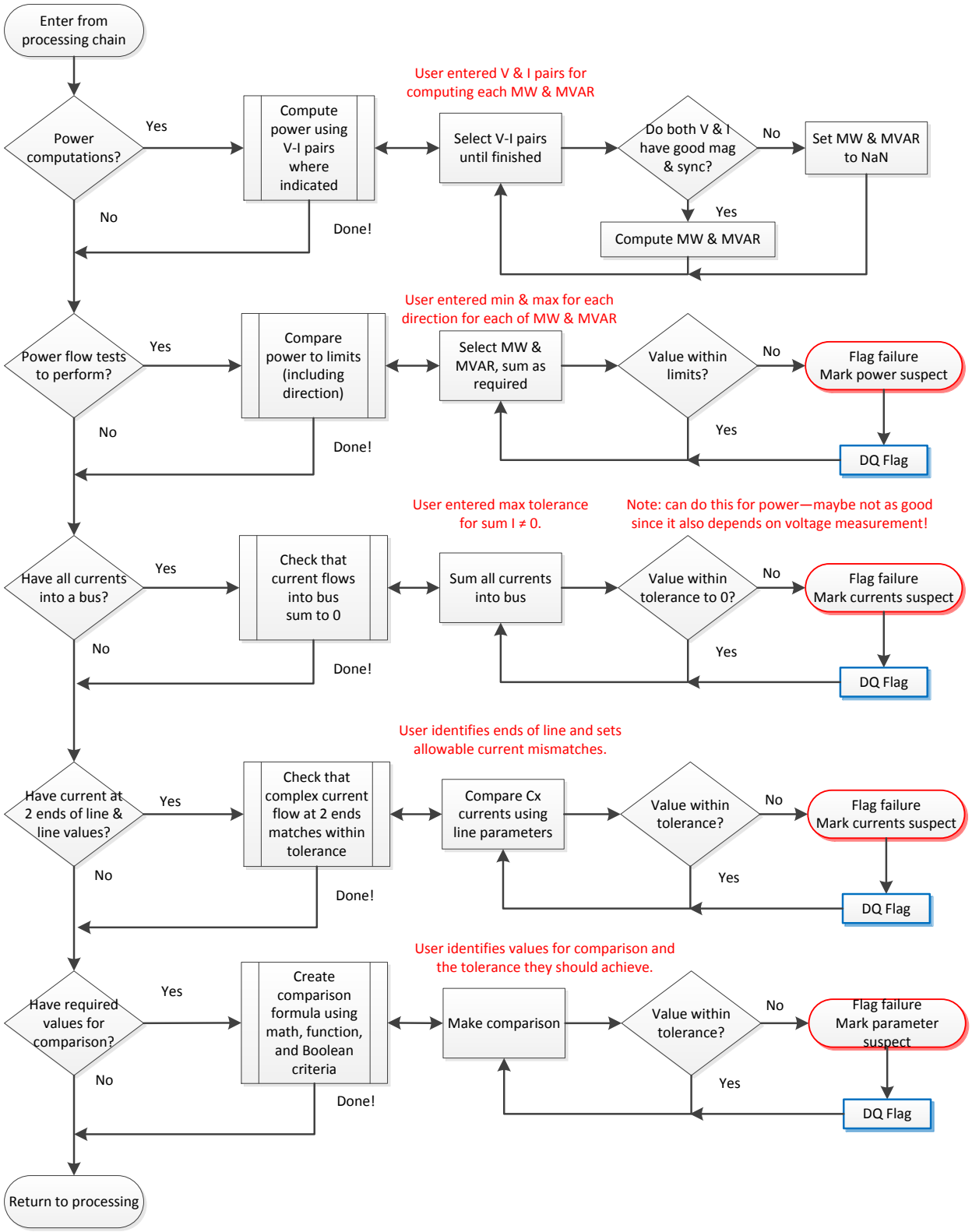

Figure 11. System Topology Processing Module

Table 6. User Enterable Function Names and Arithmetic/Logic Operators

Entry	Description
+,-,*,/, %, ^	Arithmetic operations
Abs, avg, ceil, clamp, eft, exp, floor, log, log10, log2, root, round, sqrt, sum, sign, trunc, min, max, acos, acosh, asin, ainh, atan, atan2, atanh, cos, cosh, cot, csc, sin, sinh, tan, tanh, de2rad, deg2grad, rad2deg, grad2deg	Mathematical & trigonometric operations
=, !=, <, <=, >, <=	Boolean logic
and, mand, mor, nand, nor, not, or, xor, xnor	Boolean operations
$signal_name$	Signal representation

4.11 Output processing

Output processing includes the basic communication interface necessary to connect with other equipment and support for the C37.118 messaging protocol. Most synchrophasor communications uses TCP/IP or UDP/IP over Ethernet. The interface must have provisioning for the setup of these components such as IP address and port entry, protocol selection, and so on. For the messaging contents, C37.118 allows using integer or floating point format and rectangular or polar phasor representation. It is not necessary that all these conversions be implemented, but it is preferable. Note that all data at this point is in floating point polar. Note also that if an angle is set to NaN indicating it is unusable, conversion of the phasor to rectangular form will require both components be set to NaN. In effect, the magnitude is lost. Further, conversion to integer form will almost certainly lose some resolution as there are fewer bits of precision even with the best use of scale factors. For these reasons, it is *recommended to output data in floating polar*.

The data may be output in 3 different ways:

1. It can be output just as it is received with no changes. This may be needed in some cases, but mostly defeats the purpose of this algorithm
2. It can be output with known bad data set to NaN but all uncertain data left as received. With flags this is probably the most useful data form as each application can deal with uncertain data as required for the particular use.
3. It can be output with known bad data and all uncertain data both set to NaN. This option allows using the data with the greatest confidence without having to check data flags associated with each data item.

In addition to the three ways the data itself can be presented, the quality flags for all data items can be included in the data output. The flags can be included with each of the three outputs described. In addition, the flags can be output separately but not in the C37.118 data stream format. Figure 12 illustrates the data and flag outputs.

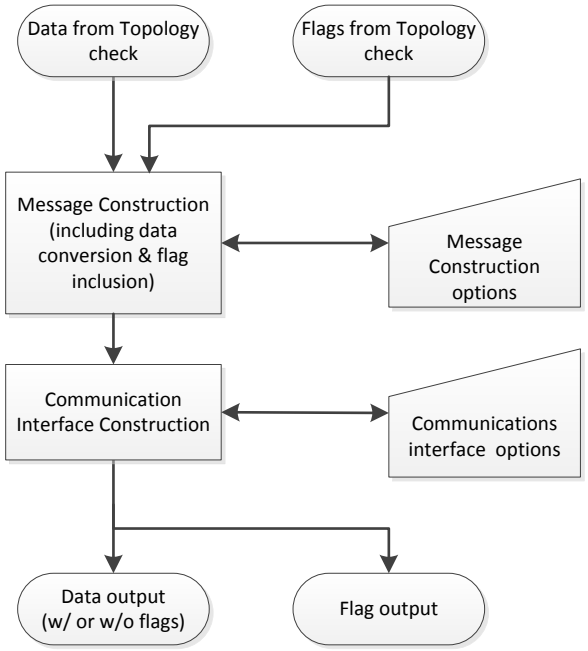

Figure 12. Message Output Processing

5. Specification Summary and Conclusion

The main object of this project is to produce an algorithm that will detect, mark, and where possible correct errors in synchrophasor data. The algorithm needs to be able to function in real time and produce a minimal delay in the data stream. It must detect certain specified errors.

This specification details an algorithm that meets these requirements. It covers all the required errors, operates in real time, and produces minimal delay. It has been demonstrated with real synchrophasor data. It is straightforward to implement and setup. This specification describes the tests to perform, the action to be taken, and construction of quality flags to indicate errors that are found. User settings are also indicated where needed. Data can be output as it was received or output with conditioning applied, both with or without quality flags. It is anticipated that a qualified software designer can implement this algorithm using this specification and relevant communication standards such as Ethernet, IP protocol, and the IEEE C37.118.2 standards.

Appendix A: Quality Code Definition, Error Flagging and Indication

The flags represent the quality state for an item's data value. This flag follows the same format as OPC DA 3.0/Field Bus standards. The design makes it easy for down-stream application to interpret the data quality without ambiguities.

The 8 bits of the Quality flags are currently defined in the form of three bit fields; Quality, Substatus and Limit status. The 8 Quality bits are arranged as follows:

```
QQSSSSLL
```

Appendix A.1 Quality Flag Bit Field

QQ	BIT VALUE	DEFINE	DESCRIPTION
0	00SSSSLL	Bad	Value is not useful for reasons indicated by the Substatus.
1	01SSSSLL	Uncertain	The quality of the value is uncertain for reasons indicated by the Substatus.
2	10SSSSLL	N/A	Not used by
3	11SSSSLL	Good	The Quality of the value is Good.

Comment:

It is recommended that clients minimally check the Quality Bit field (first 2 bits) of all results (even if they do not check the substatus or limit fields).

In several places the limit field (LL) has been used to differentiate error types in addition to the sub-status field. In those cases the specific values of the limit field are assigned. In other cases this field is simply designated LL meaning the value is ignored. It is recommended that unused fields always be set to 0 to allow simpler use if assignments are made at a later time.

The Substatus BitField:

The layout of this field depends on the value of the Quality Field.

Appendix A.2 Substatus for BAD Quality

SSSS	BIT VALUE	DEFINE	DESCRIPTION
0	000000LL	Non-specific	The value is bad but no specific reason is known. For C37.118.2 Data invalid with or without PMU error (10 or 11) is mapped to this sub-status unless the user indicates 10 indicates test mode.
1	00000100	Configuration Error	There is some server specific problem with the configuration. For example the item in question has been deleted from the configuration. This is the basic message covering non-specific errors
	00000101	Configuration Error – message format error	Specific error used in synchrophasor data processing – message is incorrectly formatted
	00000110	Configuration Error – message format error	Specific error used in synchrophasor data processing – actual message length does not match given length
	00000111	Configuration Error – message format error	Specific error used in synchrophasor data processing – PMU ID in message is incorrect (wrong destination)
2	000010LL	Not Connected	The input is required to be logically connected to something but is not. This quality may reflect that no value is available at this time, for reasons like the value may have not been provided by the data source. Dropout is mapped to this sub-status.
3	000011LL	Device Failure	A device failure has been detected.
4	000100LL	Sensor Failure	A sensor failure had been detected (the 'Limits' field can provide additional diagnostic information in some situations).
5	000101LL	Last Known Value	Communications have failed. However, the last known value is available. Note that the 'age' of the value may be determined from the timestamp.

6	00011000	Comm Failure	Communications have failed. There is no last known value is available. Default for all unspecified comm failures.
	00011001	Comm Failure - CRC error	Comm failure -- C37.118 CRC for message is in error—entire message data is unusable
	00011010		Reserved for future assignment
	00011011		Reserved for future assignment
7	000111LL	Out of Service	The block is off scan or otherwise locked. This quality is also used when the active state of the signal is InActive/Disabled. For C37.118.2 The user may indicate that PMU status bits 15/14 at 10 show test mode which is mapped to this sub-status.
8	001000LL	Waiting for Initial Data	After Items are added, it may take some time for the server to actually obtain values for these items. In such cases the client might perform a read (from cache), and/or execute a Refresh on such a subscription before the values are available.
9-15		N/A	Reserved for future use

Appendix A.3 Substatus for UNCERTAIN Quality

SSSS	BIT VALUE	DEFINE	DESCRIPTION
0	010000LL	Non-specific	There is no specific reason why the value is uncertain.
1	010001LL	Last Usable Value	Whatever was writing this value has stopped doing so. The returned value should be regarded as 'stale'. Note that this differs from a BAD value with Substatus 5 (Last Known Value). That status is associated specifically with a detectable communications error on a 'fetched' value. This error is associated with the failure of some external source to 'put' something into the value within an acceptable period of time.

2-3		N/A	Not used
4	01010000	Sensor Not Accurate	Either the value has 'pegged' at one of the sensor limits (in which case the limit field should be set to 1 or 2) or the sensor is otherwise known to be out of calibration via some form of internal diagnostics (in which case the limit field should be 0). Default for un-specific sensor errors.
	01010001	PMU Out-of-sync	Out of synch for angles is mapped to this sub-status
	01010010	SBA	Sort by arrival (local timetag)
	01010011		Reserved for future assignment
5	010101LL	Engineering Units Exceeded	The returned value is outside the limits defined for this parameter. Note that in this case the 'Limits' field indicates which limit has been exceeded but does NOT necessarily imply that the value cannot move farther out of range.
6	010110LL	Sub-Normal	The value is derived from multiple sources and has less than the required number of Good sources.
7	010111LL	Noisy	The value has high frequency noise.
8	011000LL		Reserved for future assignment
9	011001LL	Topology error	The value doesn't meet topology evaluation
10-15		N/A	Reserved for future use

Appendix A.4 Substatus for GOOD Quality

SSSS	BIT VALUE	DEFINE	DESCRIPTION
0	110000LL	Non-specific	The value is good. There are no special conditions
1-5		N/A	Not used
6	110110LL	Local Override	The value has been Overridden. Typically this is means the input has been disconnected and a manually entered value has been 'forced'

			(manual substitution).
7-15		N/A	Reserved for future use

Appendix A.5 The Limit BitField

The Limit Field is valid regardless of the Quality and Substatus. In some cases such as Sensor Failure it can provide useful diagnostic information. Low Limited and High Limited are used as part of Engineering Units Exceeded uncertain quality.

LL	BIT VALUE	DEFINE	DESCRIPTION
0	QQSSSS00	Not Limited	The value is free to move up or down
1	QQSSSS01	Low Limited	The value has 'pegged' at some lower limit
2	QQSSSS10	High Limited	The value has 'pegged' at some high limit.
3	QQSSSS11	Constant	The value is a constant and cannot move.

Appendix B: Referenced Reports

Eight reports have been created in addition to this one during this Data Validation and Conditioning project. The reports are publically available on the NASPI web site at www.naspi.org/documents. These reports are as follows:

Report 1: - FINAL Data Validation_Phase1_Task1_Review.pdf

Report 2: - FINAL Data Validation_Phase1_Task2_BestPractices.pdf

Report 3: - FINAL Data Validation_Phase1_Task3_Algorithms_Methods.pdf

Report 4: - FINAL Data Validation_Phase1_Task3_Testing_Procedures_Results.pdf

Report 5: - FINAL Data Validation_Phase1_Task4_Review_Mtgs.pdf

Report 6: - FINAL Data Validation_Phase2_Task1_Develop_Error_Simulation_Utility.pdf

Report 7: - FINAL Data Validation_Phase2_Task2_Prototype_Demonstration.pdf

Report 8: - FINAL Data Validation_Phase3_Task1_Lessons_Learned.pdf

Report 9: - FINAL Data Validation_Phase3_Task2_Functional_Specification.pdf